Apple II
Technical Notes

Developer Technical Support

UniDisk 3.5
#3: STATUS Call Bug

Revised by: Matt Deatherage November 1988
Written by: Mike Askins & Cameron Birse September 1984

This Technical Note documents a bug in the ProDOS `STATUS` call when used with a UniDisk 3.5.

The Bug

We have found that SmartPort does not return the `WRITE PROTECT` error on the `STATUS` call. (The `WRITE` call does return the `WRITE PROTECT` error as required.)

The bug manifests itself under ProDOS (and not under Pascal, since Pascal does not require the write protect error to be returned on the `STATUS` call). Specifically, if a write-protected disk is present in the UniDisk 3.5, and the application tries to write less than 512 bytes of data to a file that already exists on the media, it becomes impossible to finish the write or to close the file. Many applications ignore errors on close calls and try to reuse the buffer area which was presumably freed by the close call. This reuse results in further errors, even if the UniDisk 3.5 is later write-enabled, since ProDOS still thinks the file is open. This bug also decreases the maximum number of open files allowed, as the file left open is included in that number.

The bug also seems to cause the ProDOS `CREATE` call to fail. When a new file is created, opened and written to, and the write fails, the file manager does not deallocate the block that it reserved in the creation attempt. (The RAM copy of the bitmap seems to get trashed—`GET_FILE_INFO` calls at this point report that there are zero blocks available.) If you subsequently write **enable** the disk and do the save (with any size file), the file is written to the disk, and the bitmap is updated. The result is that there is a block reserved on the disk that no file owns, and that block cannot be freed through normal ProDOS file calls.

The Solution

Although this problem was fixed in later IIc revisions, the UniDisk 3.5 interface for the Apple][+ and IIe has never been modified. Therefore, if your application habitually performs the actions outlined above, you may avoid it by first checking to see if the media is write-protected instead of letting the buggy ProDOS `STATUS` call do it for you.

One way to accomplish this would be to issue a SmartPort `STATUS` call using a `statcode` = $00. This call returns four bytes of information, the first of which is the general status byte. This byte has the following format:

Bit	Meaning
7	0 = character device; 1 = block device
6	1 = write allowed
5	1 = read allowed
4	1 = device on line or disk in drive
3	0 = format allowed
2	**0 = medium write protected** (block devices only)
1	1 = device currently interrupting (Apple IIc only)
0	1 = device currently open (character devices only)

As shown in the table, bit 2 of this byte tells you what the ProDOS `STATUS` call cannot seem to figure out—the media in the drive is currently write-protected.